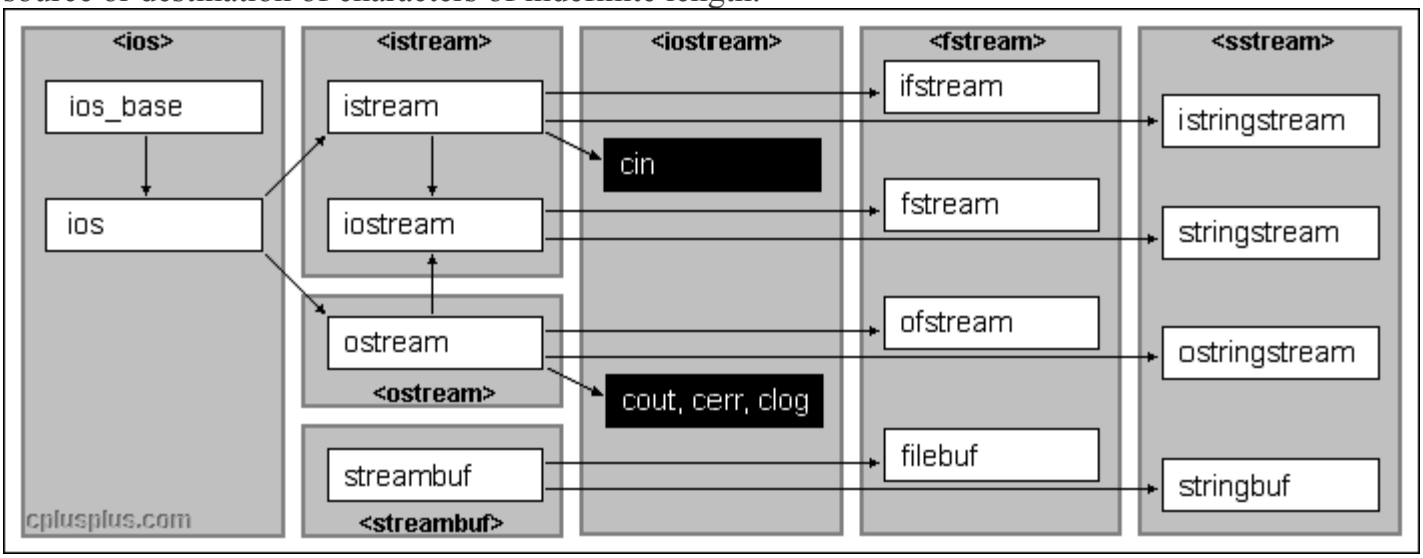
Title: Demonstrate various file operations using C++.
Objectives: 1) To learn and understand streams and files in object oriented paradigm.
2) To demonstrate file operations like create, open, read, write and close a file.
Problem Statement: Write a C++ program that creates an output file, writes information to it, closes the file and open it again as an input file and read the information from the file.
Outcomes: 1) Students will be able to learn and understand concepts of streams and files in C++.
2) Students will be able to demonstrate various operations like creating a new file, opening an existing file, reading from file, writing in file, closing file.
Hardware requirements: Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.
Software requirements: 64 bit Linux/Windows Operating System, G++ compiler
Theory:
The iostream library is an object-oriented library that provides input and output functionality using streams. A stream is an abstraction that represents a device on which input and ouput operations are performed. A stream can basically be represented as a source or destination of characters of indefinite length.



Streams are generally associated to a physical source or destination of characters, like a disk file, the keyboard, or the console, so the characters gotten or written to/from our abstraction called stream are physically input/output to the physical device. For example, file streams are C++ objects to manipulate and interact with files; Once a file stream is used to open a file, any input or output operation performed on that stream is physically reflected in the file.
So far, we have been using the iostream standard library, which provides cin and cout methods for reading from standard input and writing to standard output respectively.
This tutorial will teach you how to read and write from a file. This requires another standard C++ library called fstream, which defines three new data types:

Data Type	Description
ofstream	This data type represents the output file stream and is used to create files and to write information to files.
ifstream	This data type represents the input file stream and is used to read information from files.
fstream	This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from

files.

To perform file processing in C++, header files <iostream> and <fstream> must be included in your C++ source file.

These classes are derived directly or indirectly from the classes istream and ostream. We have already used objects whose types were these classes: cin is an object of class istream and cout is an object of class ostream. Therefore, we have already been using classes that are related to our file streams. And in fact, we can use our file streams the same way we are already used to use cin and cout, with the only difference that we have to associate these streams with physical files. Let's see an example:

```
// basic file operations
#include <iostream>
#include <fstream>
using namespace std;
int main () {
ofstream myfile;
myfile.open ("example.txt");
myfile << "Writing this to a file.\n";
myfile.close();
return 0;
}
```

Opening a File: A file must be opened before you can read from it or write to it. Either the ofstream or fstream object may be used to open a file for writing and ifstream object is used to open a file for reading purpose only.

Following is the standard syntax for open() function, which is a member of fstream, ifstream, and ofstream objects.

void open(const char *filename, ios::openmode mode);

Here, the first argument specifies the name and location of the file to be opened and the second argument of the open() member function defines the mode in which the file should be opened.

Mode Flag	Description
ios::app	Append mode. All output to that file to be appended to the end.
ios::ate	Open a file for output and move the read/write control to the end of the file.
ios::in	Open a file for reading.
ios::out	Open a file for writing.
ios::trunc	If the file already exists, its contents will be truncated before opening the file.

You can combine two or more of these values by ORing them together.For example if you want to open a file in write mode and want to truncate it in case it already exists, following will be the syntax:

ofstream outfile;

outfile.open("file.dat", ios::out | ios::trunc);

Similar way, you can open a file for reading and writing purpose as follows:

fstream afile;

afile.open("file.dat", ios::out | ios::in);

Closing a File: When a C++ program terminates it automatically closes flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.Following is the standard syntax for close() function, which is a member of fstream,

ifstream, and ofstream objects.
void close();
Writing to a File: While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen. The only difference is that you use an ofstream or fstream object instead of the cout object.
Writing operations on text files are performed in the same way we operated with cout:

```
// writing on a text file
 #include <iostream>
 #include <fstream>
 using namespace std;
 int main () {
 ofstream myfile ("example.txt");
 if (myfile.is_open())
 {
 myfile << "This is a line.\n";
 myfile << "This is another line.\n";
 myfile.close();
 }
 else cout << "Unable to open file";
 return 0;
 }
```

Reading from a File: You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an ifstream or fstream object instead of the cin object.
Reading from a file can also be performed in the same way that we did with cin:

```
// reading a text file
 #include <iostream>
 #include <fstream>
 #include <string>
 using namespace std;
 int main () {
 string line;
 ifstream myfile ("example.txt");
 if (myfile.is_open())
 {
 while ( getline (myfile,line) )
 {
 cout << line << '\n';
 }
 myfile.close();
 }
 else cout << "Unable to open file";
 return 0;
 }
```

This last example reads a text file and prints out its content on the screen. We have created a while loop that reads the file line by line, using getline. The value returned by getline is a reference to the stream object itself, which when evaluated as a boolean expression (as in this while-loop) is true if the stream is ready for more operations, and false if either the end of the file has been reached or if some other error occurred.
Text files: Text file streams are those where the ios::binary flag is not included in their opening mode. These files are designed to store text and thus all values that are input or output from/to them can suffer some formatting transformations, which do not necessarily correspond to their literal binary value.
Algorithm:

1) Include required header file like iostream (for keyboard input and output), fstream (for file reading and writing), and cstdlib (for exit()).
2) In main(), read name of file from user.
3) Open file using ofstream for writing.
4) If file does not exist, new file will be created or existing file will be overwritten.
5) If any error occurs in creating or opening file, exit
6) Otherwise, open file.
7) Read string from user.
8) Check if given termination string is entered. (Here we will use ^D to end reading contents from keyboard).
9) If ^D is entered, stop reading from keyboard.
10) Otherwise, read line from keyboard and write in file using << .
11) Close the file.
12) Open the same file for reading.
13) Check if file opened for reading or display error message and exit.
14) In while loop, start reading from file line by line and display output on the terminal.
15) If end of file is reached, exit loop.
16) End program.

Flowchart:

Draw proper flowchart and get it corrected from faculty.

Test Cases: Test Case 1: Enter name of file to create/open : try.txt Enter contents of file ending with ^D Hi Where are you going? What are you doing? Ok Bye ^D	Contents in file are : Hi Where are you going? What are you doing? Ok Bye **Test Case 2: (creating/opening a file on/from non existent path)** Enter name of file to create/open : /home/second/try.txt Error creating/opening file.

Conclusion: Hence, we have studied and learnt various file handling operations and methods available in fstream header file.

PROGRAM

```
/*
 Write a C++ program that creates an output file,
 writes information to it, closes the file and open it again as an input file
 and read the information from the file.
 Enter name of file from user
 */
 #include <cstdlib>
 #include <fstream>
 #include <iostream>
 using namespace std;
 int main ()
 {char name[80];
 cout<<"Enter name of file to create\n";
 cin>>name;
 // open a file in write mode.
 ofstream fout;
```

```
fout.open(name);
if(!fout)
{
cout<<"Error opening file\n";
exit(1);
}
cout << "Writing to the file" << endl;
cout << "Enter contents for file end with ctrl+D";
string data;
while(getline(cin,data))
{
if(data == "^D")
break;
// write inputted data into the file.
fout << data << endl;
}
// close the opened file.
fout.close();
// open a file in read mode.
ifstream fin;
fin.open(name);
if(!fin)
{
cout<<"Error opening file\n";
exit(1);
}
cout << "Reading from the file" << endl;
while(fin)
{
getline(fin,data);
// write the data at the screen.
cout << data << endl;
}
// close the opened file.
fin.close();
return 0;
}

/*
```

**********OUTPUT**********

```
[sonali@localhost ~]$ g++ -o exp expt6.cpp
[sonali@localhost ~]$ ./exp
Enter name of file to create
first.txt
Writing to the file
Enter contents for file end with ctrl+D
hi
where are you going?
what are you doing?
ok
bye
Reading from the file
hi
where are you going?
what are you doing?
ok
bye
```